Project Software Engineering (DLMCSPSE01_CF)

Inhalt

Abstract

The goal of this project is to develop an AI-based nutritional assistant. This assistant should help users improve their diet and overall health.

The assistant targets people who struggle with healthy diets because they lack the right information, or people who already have the basic knowledge but need special advice in their daily lives. Since this is a digital offer, the early adopters are digitally savvy and probably young people.

Project plan

The first milestone: the app should only support basic chat functionalities between the user and the digital assistant. To demonstrate a first more advanced feature, the first assistant tool will be implemented. The first tool will allow the assistant to store recipes in the database. With this, the first milestone will be implemented with those four epics and the scope of this assignment.

[EPIC]: Server setup epic
#53 · by djochim was closed 2 weeks ago · Duplicate · v1

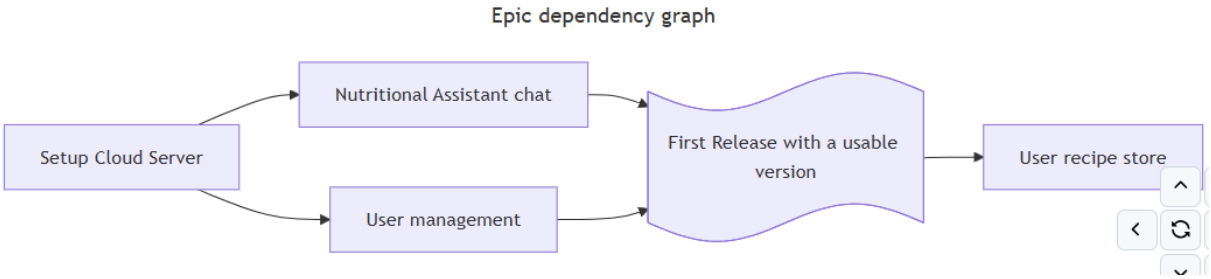
[EPIC]: User management 0 / 4 epic
#47 · djochim opened 2 weeks ago · v1

[EPIC]: Nutritional assistant chat 0 / 2 epic
#40 · djochim opened 2 weeks ago · v1

[EPIC]: User recipe store 0 / 2 epic
#39 · djochim opened 2 weeks ago · v1

[EPIC]: Setup cloud server 6 / 6 epic
#1 · by djochim was closed 6 minutes ago · v1

The following chart shows the dependencies of the epics to each other. It also shows the first usable version, that brings the first benefits to the user.



Risks

As every software project also this projects has to mitigate certain risks which can be grouped in business related risks, technological risks and security risks.

Business Risks

Data Privacy Risk

The solution experimented in this project is a 100% digital offer and the privacy concerns with those might scare off users. Especially because the assistant needs personal information about the user, to be able to support them with personalized advice.

Mitigation:

As a mitigation, data privacy should be a top priority. Users should be aware of every personal data, that is shared with the user. The User should also be able to view all data which the app collects and should be able to adapt / delete those.

In general the App should emphasize the importance of user privacy by providing good user assistance and the right data privacy features.

Product Market Risk

The problem of a healthy diet is huge and so are the solutions for it. It could be that the developed app is not a good solution to the users dietary problems. This would lead to a high user churn.

Mitigation:

The first mitigation would be a user survey, but this will not be done as part of this assignment. Instead, the first implementation will rely on the Problems of the Project member. After the first MVP is implemented, the app can be iterated based on customer feedback. This way the app can archi9ve product market fit over time.

AI Legal Risk

As outlined before, the goal is to use an AI to answer the questions to the user. The output of those generative models can only be controlled in a limited fashion. The risk is, that the AI provides some negative advice to a user.

Mitigation:

Implement clear user consent mechanism and make transparent, how generative AI works and that the recommendation's must be considered carefully and that those are no medical advices. In addition, the context definition of the assistant itself should contain guidelines to the AI, which recommendations are ok and what the AI should not do. In addition broadly adapted models should be used which are already shipped / trained with guidelines and content restrictions.

AI Wrapper Risk

An app, based on a GPT model enables problem-solving of complex problems in a limited amount of time. The risk is that at some point a generic solution also based on GPT (e.g. Chat-GPT) can do all this on their one.

Mitigation:

Always provide additional features in the specific Problem space, that other solutions are not supporting. In the first milestone the more Problem focused App visualization and the feature to store generated recipes for later use are two differentiators. In addition, the Assistant is already configured with an optimized Prompt, which a normal User will maybe not be able to write.

Technical Risks

Feature regression Risk

The software solution will be developed in multiple iterations and with small feature enhancements. It is important to prevent feature regressions, where some functionality is no-longer working as expected.

Mitigation:

Write automated tests and use CI/CD pipelines to deploy changes only when the tests are passed. The tests should be separated in low level unit tests covering critical functionalities (70% line coverage) and system tests, which test the backend service and backend client as a whole.

GPT pricing risk

The app heavily depends on GPT models which already have a certain price. If a model increases the price significantly, this impacts the operational cost of the application.

Mitigation:

In Milestone one only the GPT model of OpenAI will be connected. It must be connected using an Adapter, that will allow an easy switch to a different model at a later point in time. That way the APP could switch the Model with low effort, if the used model changes the price or another model gets cheaper or leads to better results.

Security risks

Since the app is handling critical user data, security plays an important role. All risks must be prioritized according to priority.

Unauthorized Access to Sensitive Health Data

If the User Data is not properly secured, attackers can get unauthorized access to the sensitive personal and health-related information. This can result from database misconfigurations, weak encryption or inadequate access controls.

Thread Event	Likelihood	Impact	Risk Level
Loss of Confidentiality	Possible	Significant	High
Loss of Integrity	Possible	Moderate	Moderate
Loss of Availability	Possible	Significant	High
		Overall Risk	High

Mitigations:

1. Encrypt Data at rest and in transit
2. Strict access controls
3. Secure storage using zero trust model (avoid database exposure)
4. Apply rate limiting

Data leakage from AI Models

If the Users Health Data is shared with the AI, this can imply critical data protection issues. Personal health data is very sensitive and special mechanisms must be applied.

Thread Event	Likelihood	Impact	Risk Level
Loss of Confidentiality	Likely	Moderate	Moderate
Loss of Integrity	Unlikely	Moderate	Moderate
Loss of Availability	Very Unlikely	Negligible	Low

	Overall Risk	Moderate

Mitigations:

1. Do not share sensitive health data with the assistant
2. Inform the user, that no personal health data should be shared
3. Instruct the AI model to not ask for personal health information
4. Use model with Health Insurance Portability and Accountability Act (HIPAA) support
    a. Might be important for later features
    b. Those models also allow zero retention policies

Weak or compromised Authentication

Weak authentication mechanisms or poor session management could allow unauthorized access.

Thread Event	Likelihood	Impact	Risk Level
Loss of Confidentiality	Possible	Significant	High
Loss of Integrity	Unlikely	Moderate	Moderate
Loss of Availability	Possible	Moderate	Moderate
		Overall Risk	High

Mitigations:

1. Strict password policy
2. Support multifactor authentication
3. Use short sessions and use refresh tokens for session security
4. Secure App access with biometric encryption

API Abuse and Unauthorized Data Access

Attacker can use not properly secured APIs to exploit them and to extract user data or perform unintended actions

Thread Event	Likelihood	Impact	Risk Level
Loss of Confidentiality	Possible	Moderate	Moderate
Loss of Integrity	Unlikely	Minor	Low
Loss of Availability	Possible	Moderate	Moderate
		Overall Risk	Moderate

Mitigations:

1. All APIs secured using JWT tokens
2. Rate limiting and request throttling
3. Use a web application firewall (WAF) to filter and block malicious requests
4. Strong input validation to prevent injection attacks
5. Use secure TLS versions
6. Auditlog suspicious activity

Insecure Communication Between Components

The communication between components uses Protobuf and HTTP protocols. If these communications are not secured, attackers can intercept and or manipulate data.

Ease of Attack: Medium

Thread Event	Likelihood	Impact	Risk Level
Loss of Confidentiality	Possible	Significant	Moderate

Loss of Integrity	Possible	Moderate	Moderate
Loss of Availability	Possible	Moderate	Moderate
		Overall Risk	Moderate

Mitigations:

1. Use TLS 1.3 where possible and do not allow versions lower than 1.2
2. Implement mTLS between system components
3. Apply HSTS headers to prevent protocol downgrade attacks

Software development methodology

This project will use the Kanban software development methodology. Kanban is very lightweight which makes it suitable for this small one person project. It does not require complex planning and does not force some cycles, but still allows to be flexible and to change priorities.

Kanban Board

The core focus in Kanban is the Kanban-Board. This project will be managed via Github-Projects and the Kanban board has the six columns *Backlog*, *Refined*, *Ready*, *In progress, In review* and *Done*.

The Backlog contains a list of all tasks for a bigger timeframe. It has a very big limit of 20, because in this area all items are collected for the future. Items in the Backlog are only roughly described and are not estimated and not prioritized.

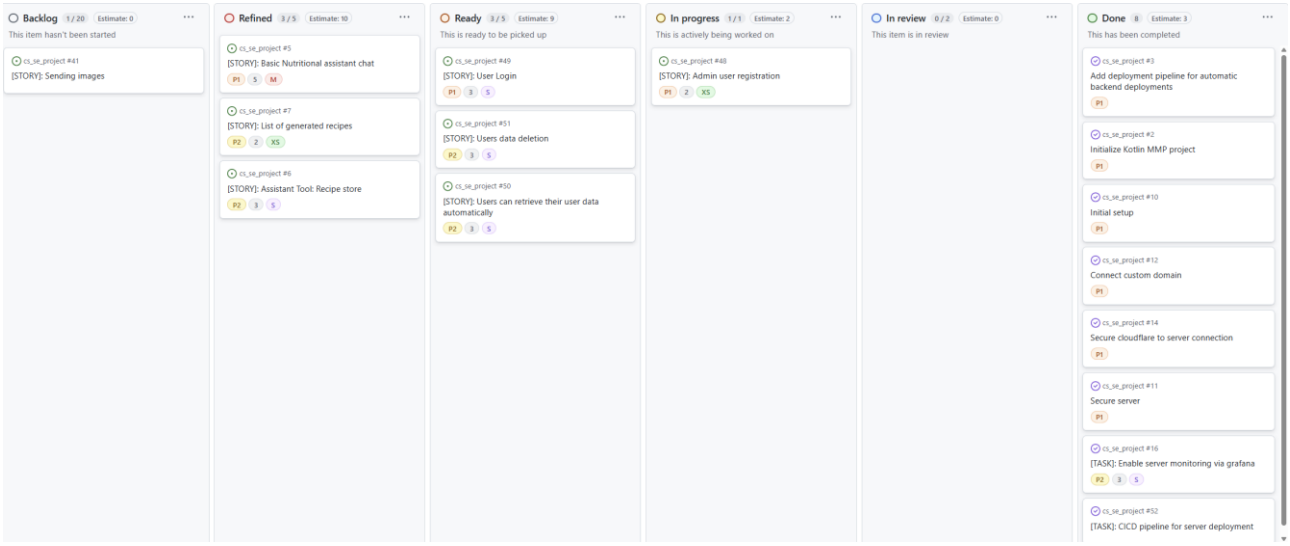
Items which are well described and which will be tackled in the next weeks, are in the Refined column. The column is limited to five prioritized and estimated items, which are ready for development as soon as all dependent requirements are fulfilled.

Once the refined items are completely unblocked, they can move in the ready column. These items could be started at any point in time. This column is also limited to five items and combined with the five refined items, they show the next up to ten items that will be done.

Started items are moved to in progress. Since this is a one-person development project, the progress column is restricted to one item. With the multitasking will be prevented.

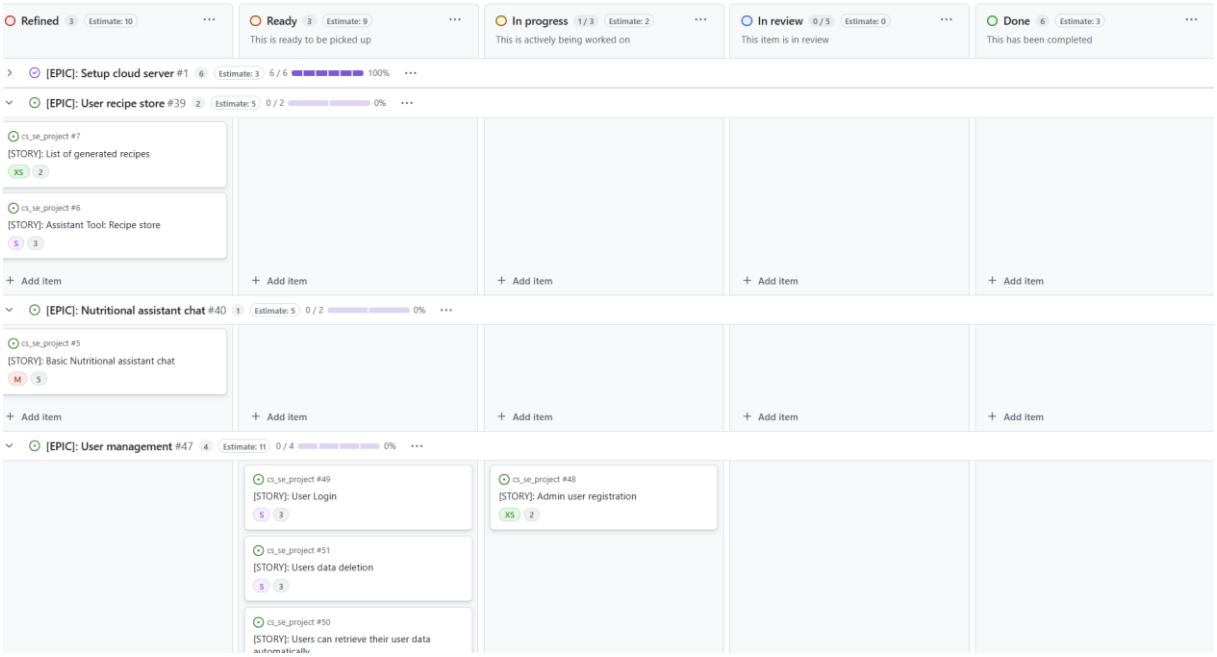
Once a PR is opened, items are automatically moved to *In review*. There are no second eyes, that could review the code, but there are automated checks, which prevent security breaches and software regressions.

After a PR is merged, items are moved to *Done*. The *Done* column has no limit configured.



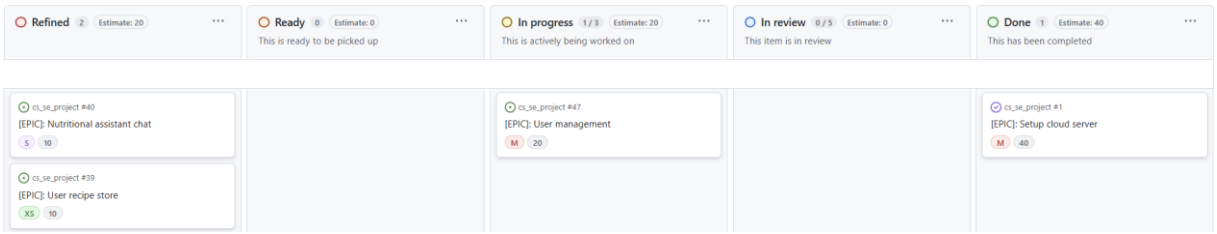
Epic Board

The epic board shows a high level overview, how far the epics have progressed and which tasks/stories there are left to complete an epic.



Milestone View

The milestone view shows the status of the overall milestone and which epics are open until the milestone is completed.



Roles

The Kanban project requires multiple roles which will all be filled with the one person in the exceptional project.

The first role is the Product Owner, who is responsible for creating the backlog and maintaining priorities based on user requirements. The Epics and User stories which the product owner defines, are further refined by the development team. This team consist of architects, developers and DevOps Engineers. The Architect is responsible for the system design and application architecture. The DevOps-Engineers create a continuous integration and continuous development pipeline. The pipeline ensures that new features are release often and only if the quality stages are passed.

Developers are responsible for the implementation of the tasks and stories. They are also responsible to write good/automated tests that ensure that new releases don’t introduce regressions and that the manual testing effort is low.

Requirements

The requirements can be grouped into non-functional and functional requirements. The non-functional requirements are required to deliver a good software solutions with high quality.

Non-functional

These non-functional requirements can be separated into quality requirements and technical requirements. The quality requirements ensure, that the software works as expected and can be used by the user. For this project the main concentration will be on software security, because the data which is handled might be critical, and it is important for the user to trust this new assistant.

The risks section already outlines the security risks and those must be mitigated by implementing security measures.

To ensure the confidentiality of the system, every communication is encrypted with TLS and communication over the public network is also using mTLS to prevent man-in-the-middle attacks of a malicious actor.

In order to be resistance under attack, the risk of Denial of service attacks must be minimized by leveraging third party prevention services. In addition, all libraries must be up-to-date and security patches applied quickly. Do ensure this, a concept of automatic updates from service libraries and the server operating system must be implemented. A static code scan should be implemented to ensure a secure development.

Besides of security measures, the functional correctness is important in order to prevent customers dissatisfaction. A unit test coverage of 70% and a basic coverage of every user story by integrated system tests ensure the functional correctness and prevent regressions.

Another quality measure to prevent customer dissatisfaction, an availability of 95% should be archived.

The service is expected to serve thousands of customer in the future. Therefore, the flexibility in terms of scalability and replaceability is important. Nevertheless, the first version only targets around 100 user with low usage rates. A concept for manual scaling and server replacement must exist.

The technical requirements ensure a fast and efficient development process and build the base for the feature development. In alignment with a lean development process, a continuous integration and deployment pipeline is required for a continuously flow of new items to the user. Next the backend server must be stetted up as a proxy to the OpenAI assistant API, but also to implement additional functionalities.

Technical requirements are maintained as tasks. Quality requirements are part of every activity (e.g. the functional correctness) or also maintained as task (e.g. the secure server task).

Every non-functional task is created via a Github Template an dcontains a basic description and acceptantce criteria.

[TASK]: CICD pipeline for server deployment task
#52 · by djochim was closed last week

[TASK]: Enable server monitoring via grafana task
#16 · by djochim was closed 4 hours ago

Secure cloudflare to server connection task
#14 · by djochim was closed on Feb 3

Connect custom domain task
#12 · by djochim was closed on Jan 29

Secure server task
#11 · by djochim was closed on Feb 15

Initial setup task
#10 · by djochim was closed on Jan 27

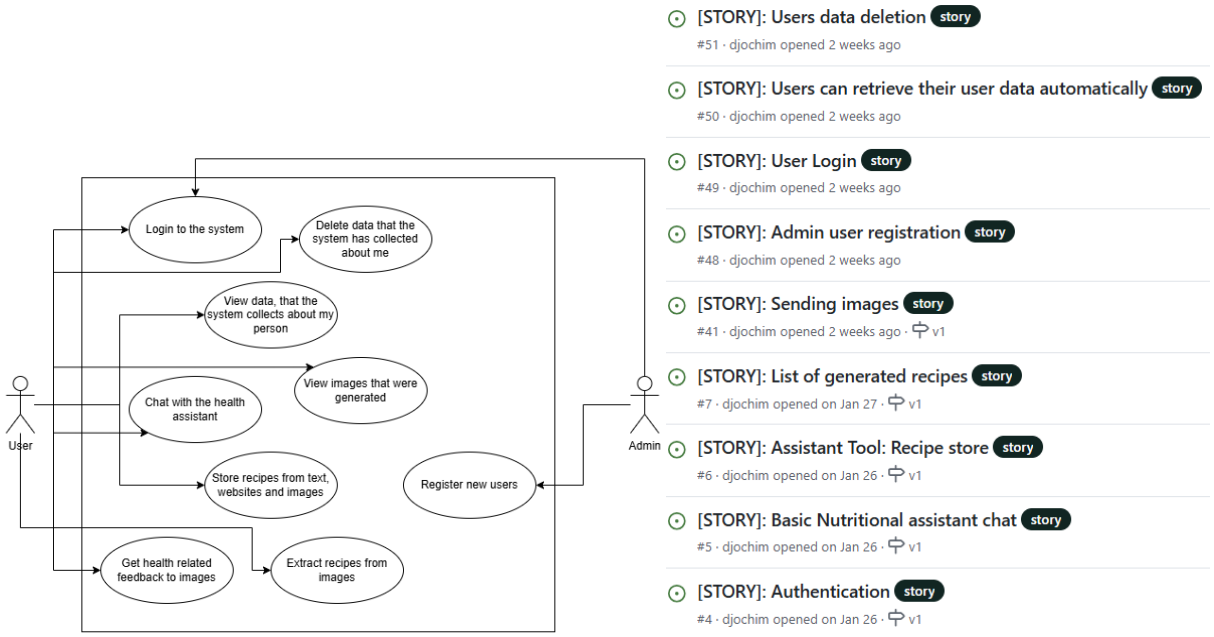
Add deployment pipeline for automatic backend deployments task
#3 · by djochim was closed 2 weeks ago · v1

Initialize Kotlin MMP project task
#2 · by djochim was closed last month · v1

Functional

For the first milestone the health coach has a very limited scope. First of all, the basic communication will be enabled between the Assistant and the user. This communication is based on text messages and the user can also send images.

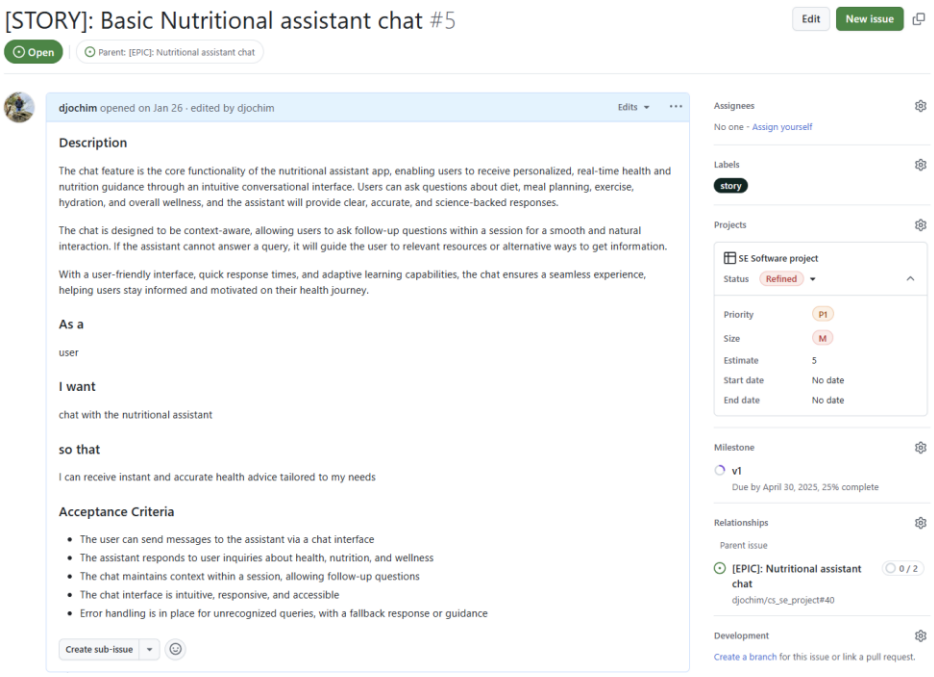
Furthermore, there will be a first toll implementation that allows the user to store recipes that are generated or extracted from text, images or websites. The use-Case-Diagram shows the stories for users and the few requirements for admins.



Feature enhancements are tracked via User Stories. User stories are created via a Template in Github and every user story contains a general description, acceptance criteria and the “As a [user/person] I want [action] so that [benefit]” phrase. Every user story is connected to a parent epic, which is grouping several related user stories. User stories can be released independently, and every story is adding a benefit for the user to the app.

Based on the status of a user story, the story is also prioritized and estimated. The priority is categorized in P0 (highest), P1 and P2 (lowest). For the estimation two values are used. The size of an item specifies the implementation effort that is required to implement something and is a good indicator how long it will take to implement an item. In contrast, the estimate number is more a definition how complex an item will be. Very complex items might require some spike to be implemented or further refinement / simplifications.

In addition, every user story is assigned to a milestone. In case of this project, every story is connected to milestone v1.



System Design

The application will consist of an Android and iOS app based on the Kotlin Multi-Platform (Jetbrains s.r.o, 2025) project. There the server, Android App and iOS APP is developed with Kotlin and all three are also able to share some of the code parts with each other. The backend server uses Ktor (Jetbrains s.r.o., 2025) as a server framework and connects to a MongoDB (MongoDB, Inc., 2025) to store required data.

The communication between the apps and the backend is based on the compact and efficient serialization format protobuf. Every request pass through Cloudflare (Cloudflare, Inc., 2025) and Ngnix (nginx, 2025) for caching and security optimizations.

Cloudflare acts as a reverse proxy and web application firewall. If protects the server against DDoS attacks and blocks traffic from known bot sites. Cloudflare is also used as the provider for the domain certificate and server certificates. In addition, Cloudflare can be used to cache static content like recipes or nutritional information of products.

The backend will be deployed on a Hetzner Cloud Server as Docker containers.

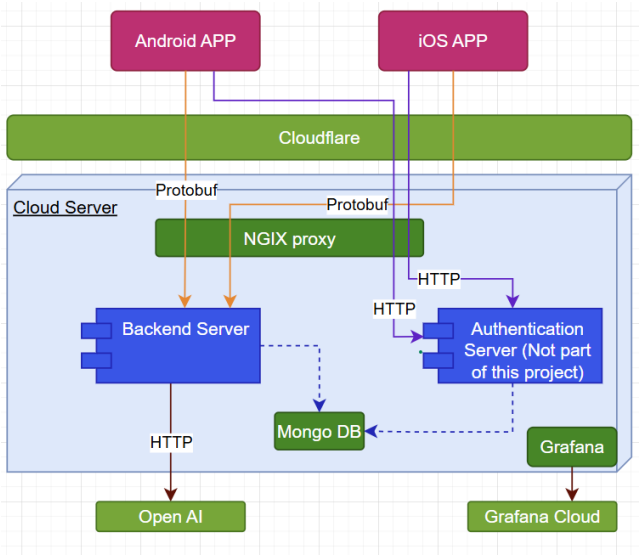
Ngnix is an additional reverse proxy inside of the server. It acts as a single entry point, only allowing traffic over the https port 433 and accepts only traffic that originates from Cloudflare.

The backend server implements the main business logic and handles the requests of users. It communicates with the public API of OpenAI (OpenAI, 2025) for the implementation of the assistant functionality.

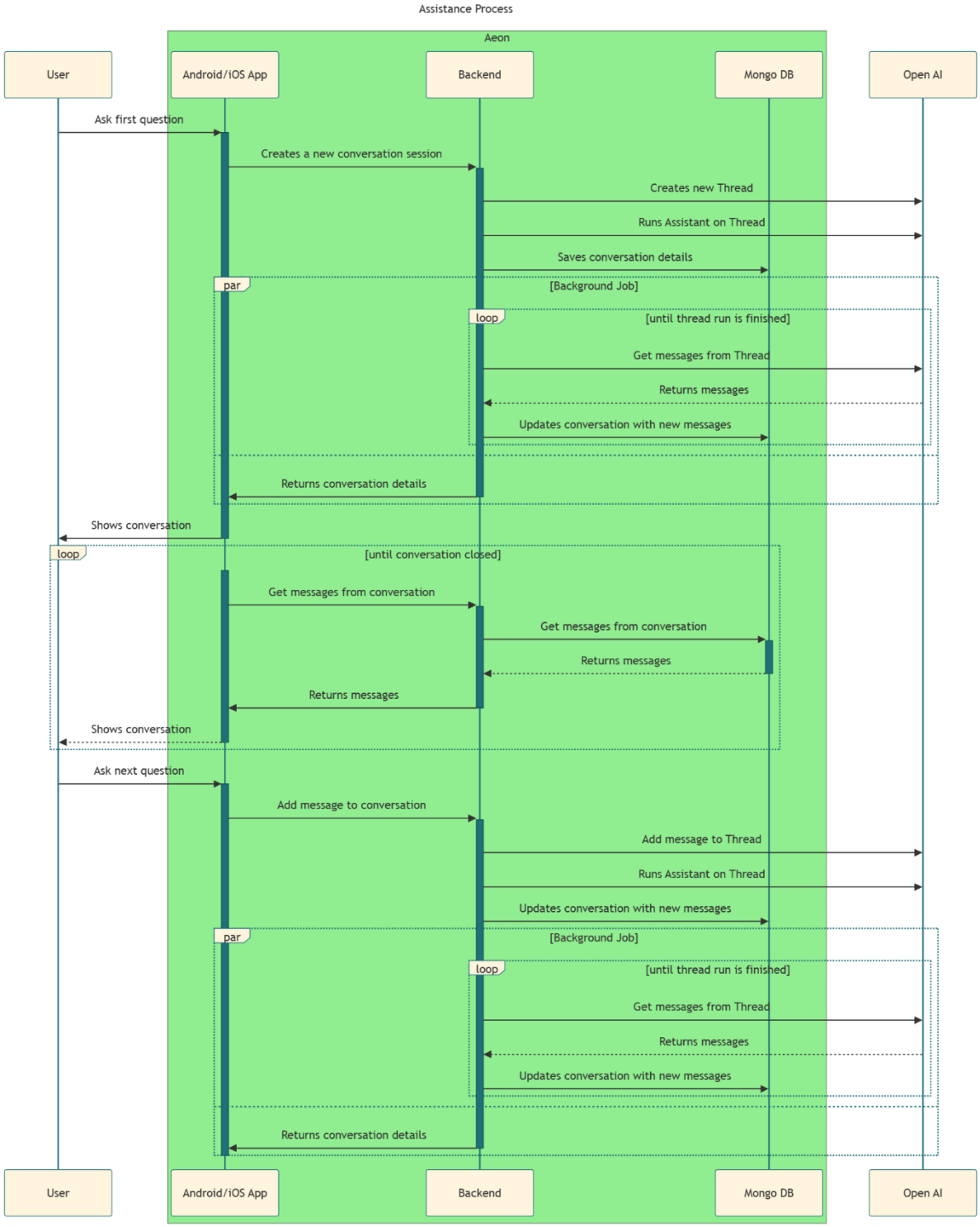
Authentication requests are handled by a separate server, which is not developed as part of this assignment.

MongoDB is used as the database, and it is used to store the chat histories, user preferences and goals and e.g. meal plans or recipes. MongoDB supports the requirement to handle unnormalized data to act as a memory for the generative AI assistant. In addition MongoDB has a good support of non-blocking clients which simplifies the integration into Ktor.

It is important to have a reliant infrastructure setup and therefore the server is monitored using Grafana (Cloudflare, Inc., 2025). For this integration a docker container with Grafana alloy is deployed inside of the server and forwards required monitoring information to the Grafana Cloud instance.



Sequence Diagram for the Assistant Chat



Development Process

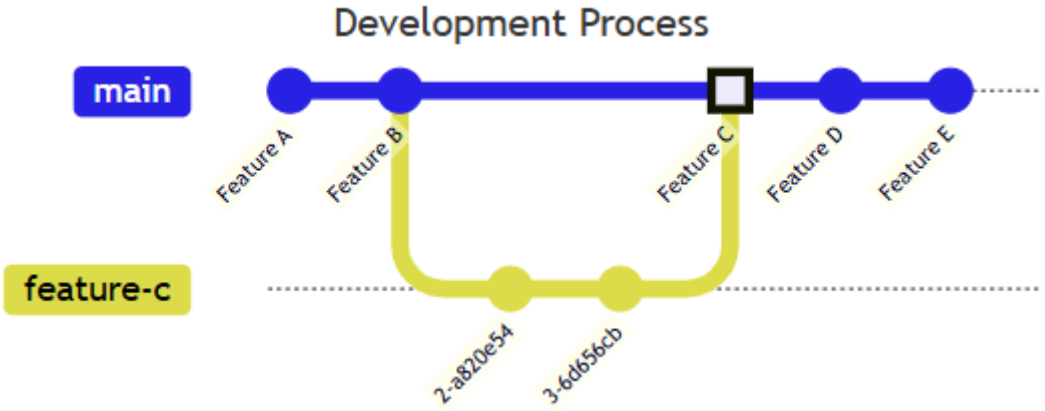
The defined development process should ensure that bug-fixes and new features are delivered to the customer as quickly as possible by ad hearing to the defined quality goals. The quick feature releases also enable quick customer feedback.

The bases of the development process is the version control system which ensures a structured development based on feature branches. GitHub is used as the tool for the version control and it integrates natively with the GitHub Project Management capabilities.

The project can be viewed publicly: djochim/cs_se_project.

For every item defined in the project Kanban board a new branch is created at the beginning of the implementation. All changes which are required are afterwards implemented on this branch and after the implementation is completed, a Pull Request must be opened.

Opening a PR (or committing changes to a PR) triggers a build pipeline and after merging the PR to the main line, the server is automatically deployed to the Hetzner-Cloud system.



Pipeline

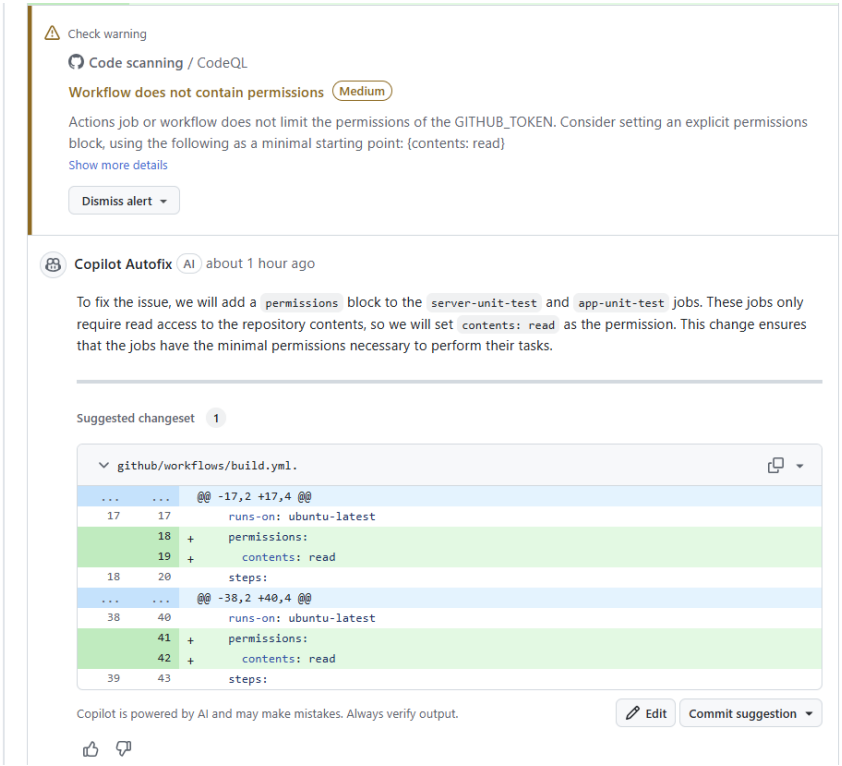
The pipeline is responsible for the automatic execution of tests and the automatic deployment of changes to the server.

The PR pipeline contains three Pipelines.

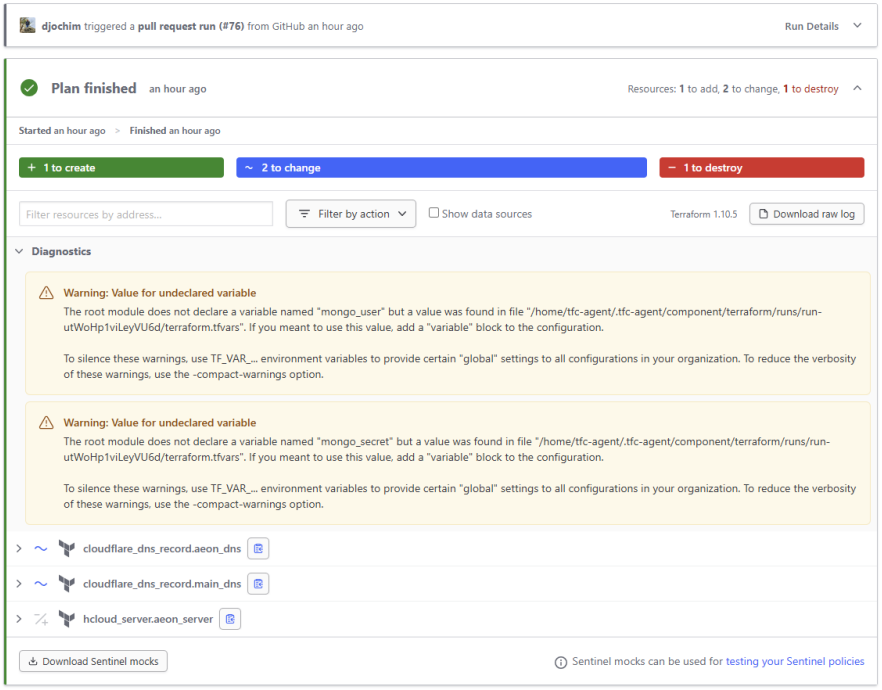
The “Build Pipeline” contains three Jobs responsible for building and testing the application. The “server-unit-test” job executes the unit tests of the server and ensures that the defined test coverage criteria for the server is met. The same is done by the “app-unit-test” job for the UI code. In addition the ”build-image” job ensures the Docker image for the server can be build.

The “CodeQL” pipeline is responsible for static security code-scans for the java code and for the pipeline security. If this pipeline as some security finding, this is directly commented in the PR as a

review, like it was in configuring the build pipeline itself:



Furthermore the PR also triggers the configured terraform project. This executes a verification of the terraform steps and prints a terraform plan to describe the required changes in order to align the configured terraform project with the actual deployed server.

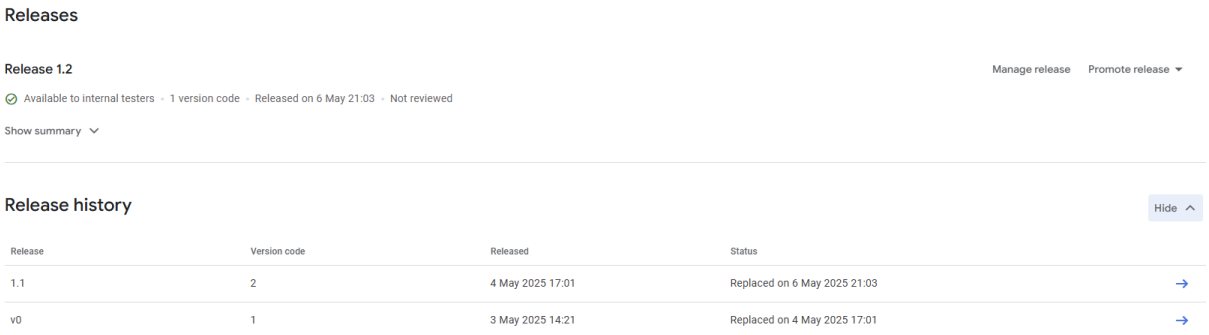


App release

The app is released manually. First locally a new version will be build and signed using an upload key of the android playstore.

This created ABB (Android App Bundle) is uploaded to the playstores internal test track. After executing manual tests to confirm the functionality of the new release, the app can be promoted to broader test tracks or production. The promotion to later tracks also requires a review of Google, that the app is following the Appstore guidelines.

Users can join the test track via this link. The playstore also shows a history of every release that was done.



Testing

As already outlined in the Pipeline section, testing is an important part of this project in order to release features which work as expected.

The big portion of tests are done via unit tests. These tests are executed automatically with every PR and also the branch and line coverage is verified, to ensure coverage of newly added code.

In addition the android project contains a few instrumented tests which are only executed manually and are running on an emulated android device and test UI components.

The app also has narrow integration tests for the clients communicating with the server. There the protobuf response of the server is mocked via the client testing tools of Ktor. These tests verify that the client is able to handle different server responses and errors correctly.

Manual End-to-End test

End-to-End test of the system is only performed manually as part of the app release process.

The manual test only covers the key functionality of the app and follows the following steps which are documented in the github repository in a test case specification.

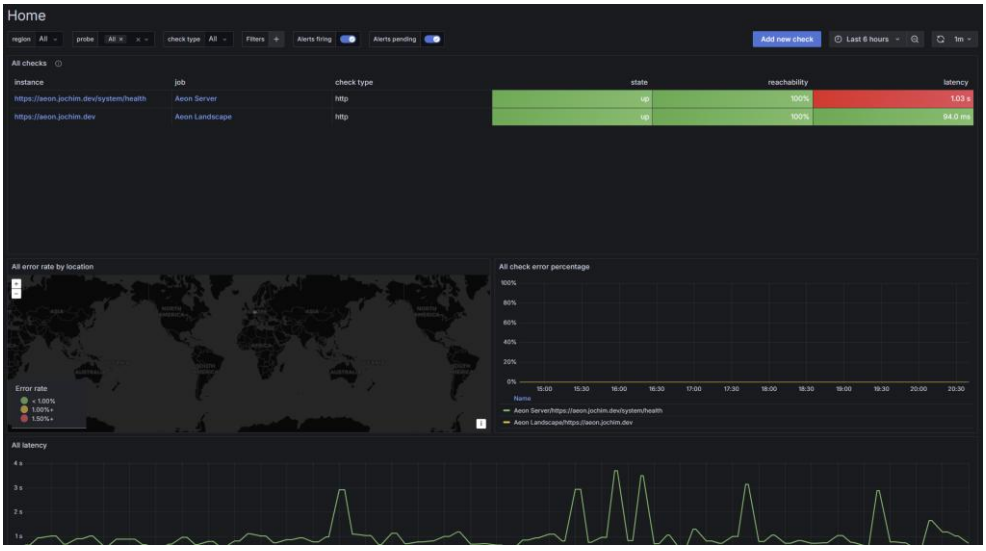
With these steps the End-to-End functionality can be ensured with the limited scope.

Observability

To ensure a running cloud system that works as expected, observability has to be integrated into the system and Landscapes must be monitored automatically.

Health Checks

The configured health checks are running regularly to verify, that the server landscape is still running and reachable. The Aeon Landscape check just pings the server and is a very quick. The Aeon Server check is more sophisticated and also verifies that dependencies of the service are reachable. This is done by querying the database and calling some endpoint of OpenAI.



Server Metrics

The server metric are retrieved directly from Hetzner Cloud. It is the source of truth in regards to server utilization.


Project aeon
Servers All
Load Balancers All
Last 30 minutes
Refresh
Time range zoom out
CTRL+Z
Servers
CPU
40%
30%
20%
10%
0%
20:25
20:30
20:35
20:40
20:45
20:50
aeon-server
Disk Bandwidth
0 B/s
-128 KiB/s
-256 KiB/s
-384 KiB/s
-512 KiB/s
Read aeon-server
Write aeon-server
Disk Bandwidth
0 io/s
-20 io/s
-40 io/s
-60 io/s
-80 io/s
Read aeon-server
Write aeon-server
Network Bandwidth
1.50 KiB/s
1 KiB/s
512 B/s
0 B/s
-512 B/s
-1 KiB/s
Received aeon-server
Sent aeon-server
Network PPS
10 p/s
5 p/s
0 p/s
-5 p/s
-10 p/s
Received aeon-server
Sent aeon-server

Glossary

Chat Interaction	The communication between the user and the assistant via a Chat interface
Nutritional assistant	The assistant that provides dietary recommendations to the user
Recipe	A plan for cooking a meal based on ingredients and steps

References

Cloudflare, Inc. (2025, 05 10). *Cloudflare*. Retrieved from https://www.cloudflare.com/

Docker Inc. (2025, 05 10). *Docker*. Retrieved from https://www.docker.com/

Grafana Labs. (2025, 05 10). *Grafana*. Retrieved from https://grafana.com/

Hetzner Online GmbH. (2025, 05 10). *Hetzner Cloud*. Retrieved from https://www.hetzner.com/de/cloud/

Jetbrains s.r.o. (2025, 05 08). *Kotlin Multiplatform*. Retrieved from https://www.jetbrains.com/compose-multiplatform/

Jetbrains s.r.o. (2025, 12 10). *Ktor*. Retrieved from https://www.jetbrains.com/compose-multiplatform/

MongoDB, Inc. (2025, 05 10). *MongoDB*. Retrieved from https://www.mongodb.com/

nginx. (2025, 05 10). *nginx*. Retrieved from https://nginx.org/

OpenAI. (2025, 05 10). *OpenAI API*. Retrieved from https://openai.com/api/